Tópicos em Estatística 2 **Prof.:** George von Borries

1 de julho de 2024

Bagging em Árvores de Decisão:

Melhorando a estabilidade e acurácia de modelos preditivos

Grupo 3
Joao Victor de Oliveira Nogueira
Pedro Henrique Lima de Menezes
Rafael Costa Ramos
Vinicius Martins Tostes

O que são ensembles?

O *ensemble* é uma técnica de machine learning que combina múltiplos modelos de classificação ou regressão buscando aumentar a estabilidade e acurácia para além de qualquer modelo individual que o compõe. A motivação é a mesma da ideia de que os votos ou palpites de um grupo de pessoas serão em geral melhores que os votos ou palpites de uma dessas pessoas apenas (sabedoria das multidões).

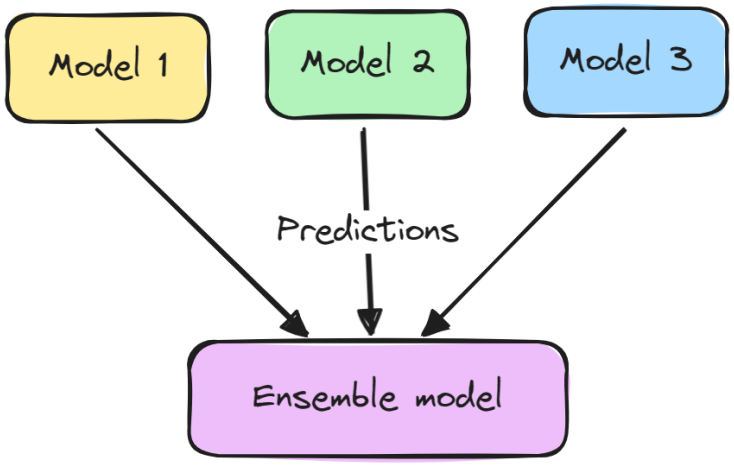

Figura 1: Ilustração de ensemble (DataCamp).

Dentre os algoritmos mais populares de ensemble estão o *bagging*, o *boosting* e o *stacking*. Nesta apresentação falaremos acerca do bagging.

O que é o Bagging?

O Bagging é um algoritmo de ensemble que agrega múltiplas *versões* de um mesmo preditor para aumentar a estabilidade e a acurácia em problemas de regressão e classificação. Sua importância é reduzir a variância de estimativas, aumentando a sua estabilidade, e evitar o *overfitting*.

Por que fazer Bagging?

Apesar de podermos aplicar o bagging sobre qualquer método preditivo, ele é particularmente útil para métodos instáveis, cujas estimativas variam muito de ajuste para ajuste. E esse é o caso das árvores de decisão.

A natureza de particionamento das árvores de decisão fornece flexibilidade e capacidade de ajuste ilimitados a elas. Isto é, se dermos a cada observação seu próprio nó, é possível obter o ajuste "perfeito". Porém isso vem a dois custos principais:

(i) Árvores de decisão facilmente podem não generalizar bem para fora do conjunto de treinamento (*overfitting*) se não ajustadas e validadas com cuidado;

(ii) Alteração mínimas no conjunto de dados podem resultar em árvores completamente diferentes (instabilidade).

Além das árvores de decisão, Breiman (1996) traz em seu paper original um outro exemplo de aplicação do Bagging envolvendo métodos de seleção de preditores (*best subset selection*, *stepwise selection*, etc.) em regressão linear, que também são procedimentos instáveis.

O algoritmo de Bagging

O meta-algoritmo de Bagging proposto por Breiman (1996) consiste em ajustar um mesmo método preditivo a diferentes reamostras de bootstrap do conjuntos de dados.

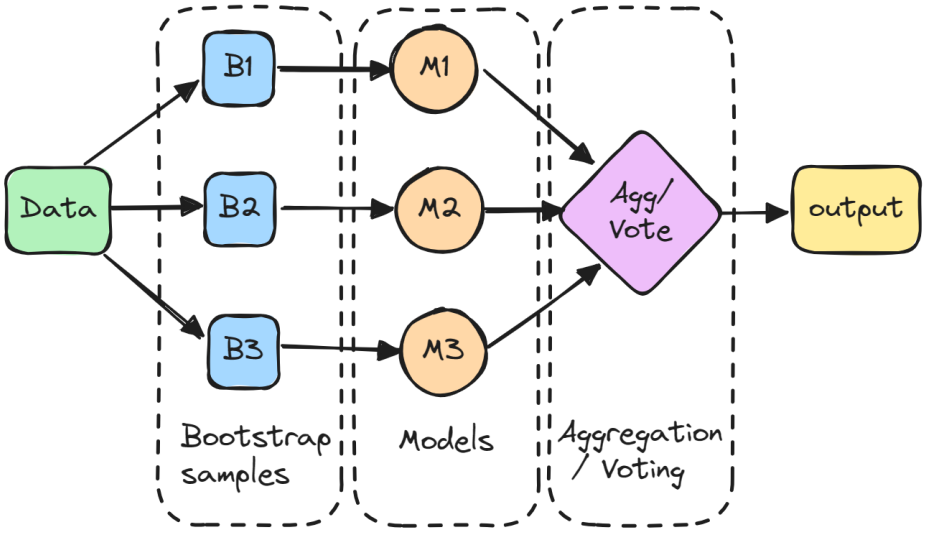

Figura 2: O algoritmo de Bagging (DataCamp).

Seja $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$ um conjunto de dados de tamanho n composto por pares de preditores x e variável resposta y. Considere ainda $\{\mathcal{D}_b\}_{b=1}^B$ um conjunto de B diferentes amostras aleatórias, de mesmo tamanho n, com reposição (boostrap) do conjunto $\mathcal{D}$.

Assim, para cada amostra $\mathcal{D}_b$ ajusta-se um preditor $\hat{y}_b(\mathbf{x}) = \hat{y}(\mathbf{x}; \mathcal{D}_b)$. Com isso, as predições via bagging são obtidas com base na agregação das predições individuais $\{\hat{y}_b(\mathbf{x})\}_{b=1}^B$ de todos

esses modelos. No caso de regressão, a agregação tipicamente é feita pela média,

$$\hat{y}_{\text{bag}}(\mathbf{x}) = \frac{1}{B}\sum_{b=1}^{B} \hat{y}_b(\mathbf{x}),$$

e em classificação, a classe mais "votada" (moda),

$$\hat{y}_{\text{bag}}(\mathbf{x}) = \arg\max_{c} \#\{b\colon \hat{y}_b(\mathbf{x}) = c\}.$$

Denote por $(\mathcal{D} - \mathcal{D}_b)$ as observações do conjunto de dados que não foram selecionadas na b-ésima amostra bootstrap. Essas observações são chamadas de *out-of-bag* (OOB) e podem ser usadas como dados de teste, dispensando o uso de validação cruzada. A média dos erros médios dos modelos em suas respectivas amostras OOBs é chamado de erro OOB.

Bishop (2006) traz, para o caso de regressão, um argumento matemático/estatístico de que esse procedimento de bagging garantidamente reduz o erro de predição médio, embora a redução possa ser pequena.

Outros usos e formas de bagging

Ainda no contexto de árvores de decisão, o bagging pode ser aplicado também em outro sentido, ou melhor, em outra dimensão. Quando o bagging é feito não só no sentido das linhas/observações do conjunto de dados, mas também no sentido das colunas/preditores, temos o que seriam as chamadas florestas aleatórias (*random forests*). Esse tema será abordado pelo grupo seguinte.

Exemplo: Câncer de mama

Apresentaremos um exemplo de bagging em árvores de decisão utilizando um conjunto de dados sobre câncer de mama. O objetivo é identificar, com base em suas características, se tumores de mama são malignos ou benignos, isto é, se as células anormais têm tendência a se espalhar para outras partes do corpo ou não.

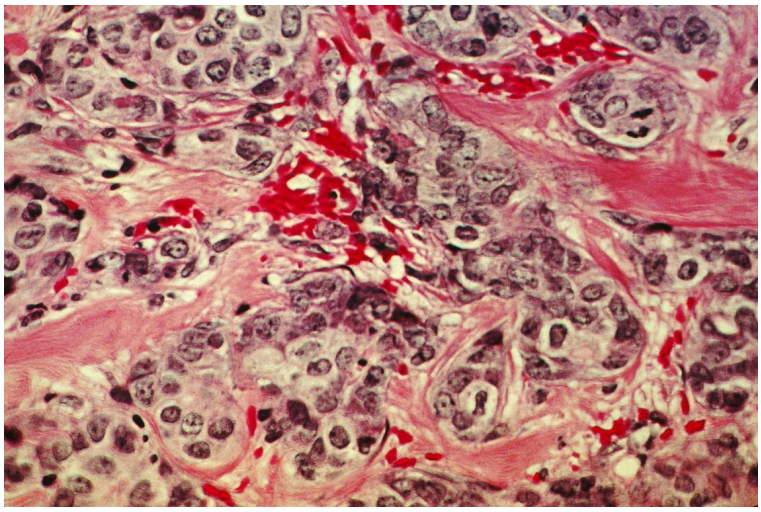

Figura 3: Células de câncer de mama (Duke Today).

As trinta variáveis preditoras do conjunto foram computadas a partir de uma imagem microscópica de tecido extraído do tumor via punção aspirativa por agulha fina (PAAF). Elas descrevem características dos núcleos de células presentes na imagem (raio, textura, perímetro, área, suavidade, compacidade, concavidade, pontos de concavidade, simetria e dimensão fractal), resumidas em média (*mean*), erro-padrão (*se*) e pior (*worst*, a média dos três maiores valores).

```
library(tidyverse)
breast_cancer <- read_csv("breast_cancer.csv")
breast_cancer <- breast_cancer |>
  select(-id, -...33) |>
  janitor::clean_names()

glimpse(breast_cancer)
```

```
Rows: 568
Columns: 31
$ diagnosis               <chr> "M", "M", "M", "M", "M", "M", "M", "M", "M", "~
$ radius_mean             <dbl> 17.990, 20.570, 19.690, 11.420, 20.290, 12.450~
$ texture_mean            <dbl> 10.38, 17.77, 21.25, 20.38, 14.34, 15.70, 19.9~
$ perimeter_mean          <dbl> 122.80, 132.90, 130.00, 77.58, 135.10, 82.57, ~
$ area_mean               <dbl> 1001.0, 1326.0, 1203.0, 386.1, 1297.0, 477.1, ~
$ smoothness_mean         <dbl> 0.11840, 0.08474, 0.10960, 0.14250, 0.10030, 0~
$ compactness_mean        <dbl> 0.27760, 0.07864, 0.15990, 0.28390, 0.13280, 0~
$ concavity_mean          <dbl> 0.30010, 0.08690, 0.19740, 0.24140, 0.19800, 0~
$ concave_points_mean     <dbl> 0.14710, 0.07017, 0.12790, 0.10520, 0.10430, 0~
$ symmetry_mean           <dbl> 0.2419, 0.1812, 0.2069, 0.2597, 0.1809, 0.2087~
$ fractal_dimension_mean  <dbl> 0.07871, 0.05667, 0.05999, 0.09744, 0.05883, 0~
$ radius_se               <dbl> 1.0950, 0.5435, 0.7456, 0.4956, 0.7572, 0.3345~
```

```
$ texture_se              <dbl> 0.9053, 0.7339, 0.7869, 1.1560, 0.7813, 0.8902~
$ perimeter_se            <dbl> 8.589, 3.398, 4.585, 3.445, 5.438, 2.217, 3.18~
$ area_se                 <dbl> 153.40, 74.08, 94.03, 27.23, 94.44, 27.19, 53.~
$ smoothness_se           <dbl> 0.006399, 0.005225, 0.006150, 0.009110, 0.0114~
$ compactness_se          <dbl> 0.049040, 0.013080, 0.040060, 0.074580, 0.0246~
$ concavity_se            <dbl> 0.05373, 0.01860, 0.03832, 0.05661, 0.05688, 0~
$ concave_points_se       <dbl> 0.015870, 0.013400, 0.020580, 0.018670, 0.0188~
$ symmetry_se             <dbl> 0.03003, 0.01389, 0.02250, 0.05963, 0.01756, 0~
$ fractal_dimension_se    <dbl> 0.006193, 0.003532, 0.004571, 0.009208, 0.0051~
$ radius_worst            <dbl> 25.38, 24.99, 23.57, 14.91, 22.54, 15.47, 22.8~
$ texture_worst           <dbl> 17.33, 23.41, 25.53, 26.50, 16.67, 23.75, 27.6~
$ perimeter_worst         <dbl> 184.60, 158.80, 152.50, 98.87, 152.20, 103.40,~
$ area_worst              <dbl> 2019.0, 1956.0, 1709.0, 567.7, 1575.0, 741.6, ~
$ smoothness_worst        <dbl> 0.1622, 0.1238, 0.1444, 0.2098, 0.1374, 0.1791~
$ compactness_worst       <dbl> 0.6656, 0.1866, 0.4245, 0.8663, 0.2050, 0.5249~
$ concavity_worst         <dbl> 0.71190, 0.24160, 0.45040, 0.68690, 0.40000, 0~
$ concave_points_worst    <dbl> 0.26540, 0.18600, 0.24300, 0.25750, 0.16250, 0~
$ symmetry_worst          <dbl> 0.4601, 0.2750, 0.3613, 0.6638, 0.2364, 0.3985~
$ fractal_dimension_worst <dbl> 0.11890, 0.08902, 0.08758, 0.17300, 0.07678, 0~
```

Os tumores do conjunto de dados estão divididos em 63% de casos malignos (M) e 37% de casos benignos (B).

```
breast_cancer |>
  group_by(diagnosis) |>
  summarise(n = n()) |>
  mutate(frac = n / sum(n))
```

```
# A tibble: 2 x 3
  diagnosis     n  frac
  <chr>     <int> <dbl>
1 B           356 0.627
2 M           212 0.373
```

Árvore de decisão

Usando o pacote `caret`, podemos ajustar uma árvore de decisão por meio do método `rpart`. Para ajustar a árvore, precisamos definir o parâmetro de complexidade `cp`, que dita se "vale a pena" fazer um split em um dado nó da árvore. Em síntese, valores próximos de zero facilitam os splits e valores próximos de um dificultam.

Para ajustar o parâmetro `cp`, nos baseamos na validação cruzada 10-*fold*. Os valores experimentados e as respectivas acurácia de classificação obtidas de validação são exibidos logo em seguida e na Figura 4. Nesse caso, o valor ótimo foi `cp = 0`, que trouxe uma acurácia de 93%.

```r
set.seed(42)

library(caret)
rpart_fit <- train(
  diagnosis ~ .,
  data = breast_cancer,
  method = "rpart",
  # Validação cruzada 10-fold
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10,
)
rpart_fit
```

```
CART

568 samples
 30 predictor
  2 classes: 'B', 'M'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 512, 511, 511, 510, 511, 512, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.00000000  0.9313877  0.8527760
  0.08805031  0.8997440  0.7832704
  0.17610063  0.8961725  0.7749746
  0.26415094  0.8961725  0.7749746
  0.35220126  0.8961725  0.7749746
  0.44025157  0.8961725  0.7749746
  0.52830189  0.8961725  0.7749746
  0.61635220  0.8961725  0.7749746
  0.70440252  0.8961725  0.7749746
  0.79245283  0.7771249  0.4393609

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.
```

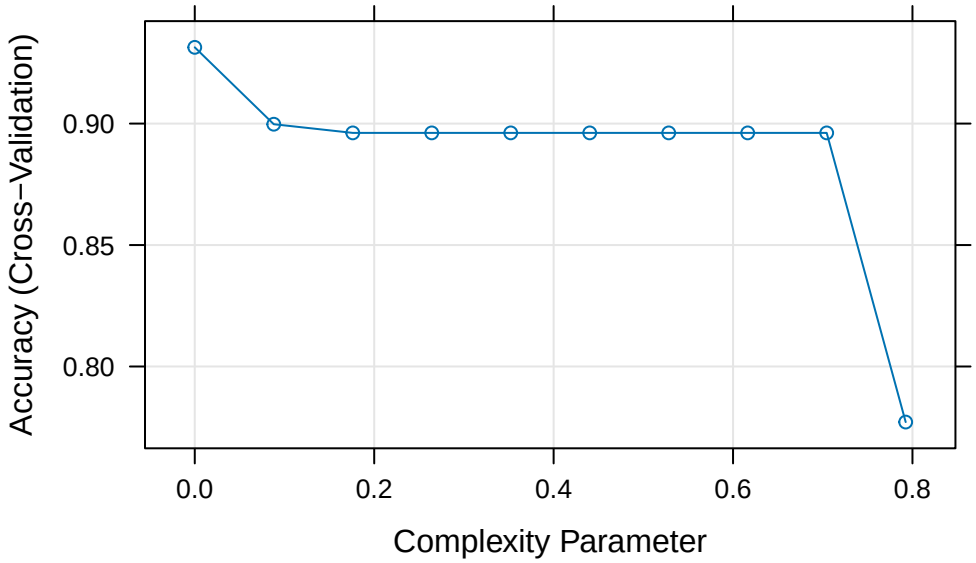

Figura 4: Acurácia de validação da árvore de decisão em função do parâmetro de complexidade (cp).

A Figura Figura 5 apresenta a árvore de decisão escolhida na validação cruzada. Em cada nó temos o tipo de câncer predito (mais provável) até aquele nó, a proporção de casos malignos (M) e a porcentagem de pacientes no nó.

```
library(rpart.plot)
rpart.plot(rpart_fit$finalModel)
```

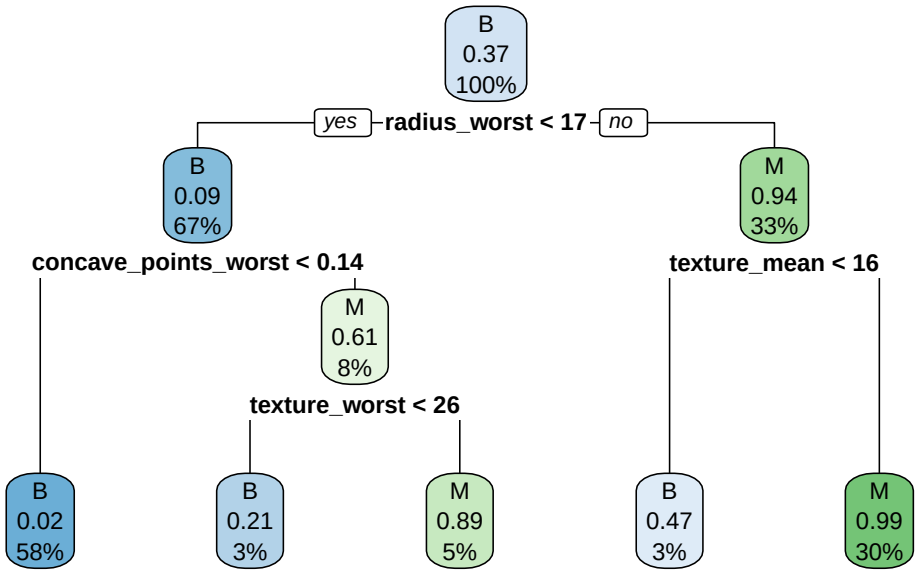

Figura 5: Árvore de decisão para classificação dos tumores.

Agora, para ilustrar a instabilidade das árvores de decisão, vamos retirar um subconjunto aleatório pequeno de 5% das observações de treinamento e ver o que acontece com o resultado. Isso é feito duas vezes e os resultados são mostrados na Figura 6.

```
set.seed(42)

par(mfrow = c(1, 2))
for(i in 1:2){
  rpart_fit2 <- train(
    diagnosis ~ .,
    # Remoção de 5% dos dados
    data = slice_sample(breast_cancer, n = 540, replace = FALSE),
    method = "rpart",
    # Validação cruzada 10-fold
    trControl = trainControl(method = "cv", number = 10),
    tuneLength = 10,
  )
  rpart.plot(rpart_fit2$finalModel)
}
```

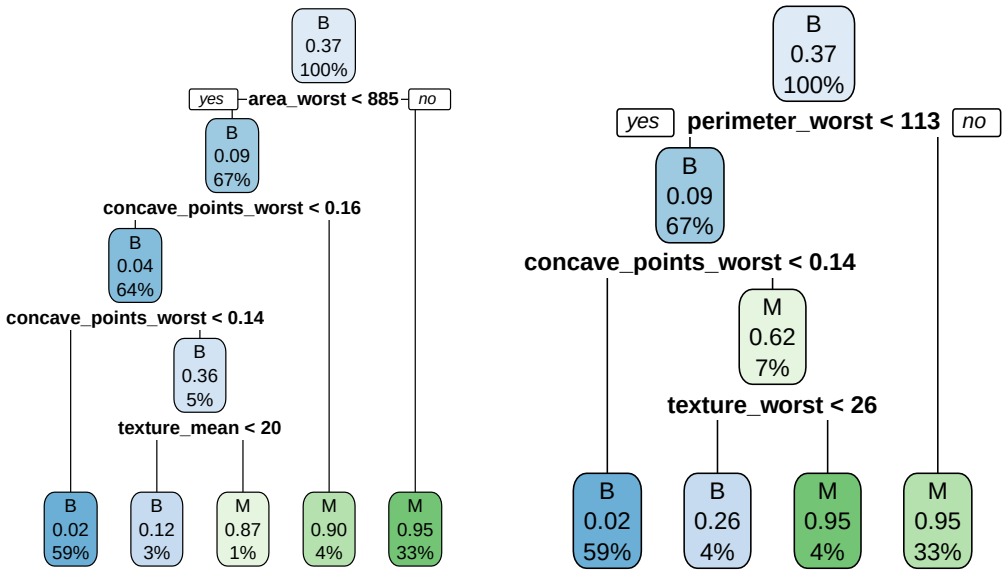

Figura 6: Árvore de decisão para classificação dos tumores, após retirar cinco porcento das observações.

Veja que após a retirada de poucas observações temos árvores completamente diferentes da primeira, desde o nó inicial. Não só temos diferentes variáveis nos splits, como as árvores possuem diferentes profundidades.

Bagging de árvores de decisão

Agora vamos experimentar o conceito de bagging nas árvores de decisão. Novamente usaremos o pacote `caret`. Mas, antes de tudo, os dados serão particionados em treino (80%) e teste (20%), para que ao final possamos testar o modelo.

```
library(caret)
library(rpart)

set.seed(42)
in_train <- createDataPartition(breast_cancer$diagnosis, p = .8,
                                list = FALSE,
                                times = 1)

breast_train <- breast_cancer[in_train, ]
breast_test <- breast_cancer[-in_train, ]
```

Agora podemos ajustar as àrvores com bagging usando o método `treebag` do pacote `caret`. Seguindo os resultados obtidos anteriormente, definimos os parâmetros das árvores como

`minsplit = 2` e `cp = 0`. A Figura Figura 7 mostra as acurácias obtidas na validação cruzada 10-fold e no out-of-bag para diferentes números de árvores no bagging.

```
set.seed(42)
nbagg <- seq(10, 50, by = 10)

tb_nbagg_cv <- map(nbagg, ~ {
  tb_fit <- train(
    diagnosis ~ .,
    data = breast_train,
    method = "treebag",
    # Validação cruzada 10-fold
    trControl = trainControl(method = "cv", number = 10),
    # Número de árvores
    nbagg = .x,
    # Parâmetros das árvores, ver help(rpart.control)
    control = rpart.control(minsplit = 2, cp = 0)
  )
  tb_fit$results
})

tb_nbagg_oob <- map(nbagg, ~ {
  tb_fit <- train(
    diagnosis ~ .,
    data = breast_train,
    method = "treebag",
    # Validação out-of-bag
    trControl = trainControl(method = "none"),
    coob = TRUE,
    keepX = TRUE,
    # Número de árvores
    nbagg = .x,
    # Parâmetros das árvores
    control = rpart.control(minsplit = 2, cp = 0)
  )
  tb_code <- getModelInfo("treebag")[[1]]
  tb_code$oob(tb_fit$finalModel)
})
```

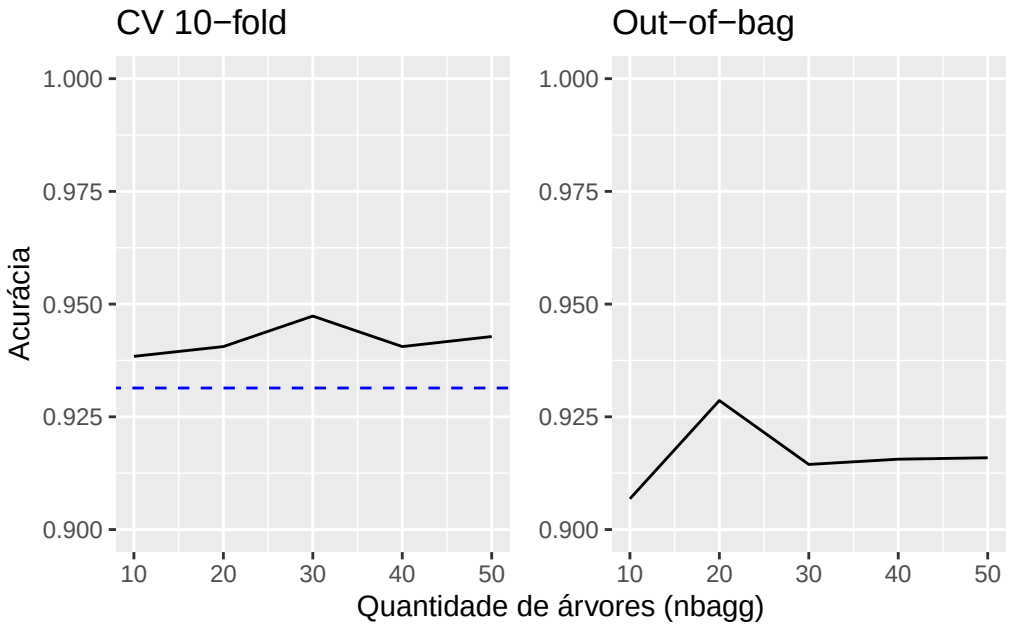

Figura 7: Acurácia de validação cruzada 10-fold e OOB da árvore de decisão com bagging em função do tamanho do ensemble (número de árvores).

Note que, para todos os números de árvores testados, as acurácias no CV das árvores de decisão com bagging são melhores que a acurácia da árvore de decisão simples (linha pontilhada azul) que testamos inicialmente. Vemos ainda que, nesse caso, a acurácia não muda muito com o número de árvores `nbagg`, e que 30 árvores já são suficientes para uma acurácia de 94%, um pouco maior que o árvore de decisão simples.

```
set.seed(42)

treebag_fit <- train(
  diagnosis ~ .,
  data = breast_train,
  method = "treebag",
  # Validação cruzada 10-fold
  trControl = trainControl(method = "cv", number = 10),
  # Número de árvores
  nbagg = 30,
  # Parâmetros das árvores, ver help(rpart.control)
  control = rpart.control(minsplit = 2, cp = 0)
)
treebag_fit
```

```
Bagged CART
```

```
455 samples
 30 predictor
  2 classes: 'B', 'M'

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 410, 409, 410, 410, 409, 410, ...
Resampling results:

  Accuracy   Kappa
  0.9384541  0.8668773
```

De modo geral, os resultados melhorarão conforme se aumenta o número de árvores, mas rapidamente o ganho se torna insignificante frente ao custo computacional. Felizmente, embora o número ótimo de árvores dependa do conjunto de dados, dificilmente serão necessárias mais que 100 árvores para obter um bom resultado. No pacote `caret`, inclusive, o padrão é `nbagg = 25`.

A matriz de confusão a seguir mostra que, no conjunto de treinamento, apenas 1 dos 455 (0,2%) tumores foi classificado incorretamente.

```
diagnosis_pred <- predict(treebag_fit)
xtabs(~ diagnosis_pred + diagnosis, data = breast_train)
```

```
              diagnosis
diagnosis_pred   B   M
             B 285   1
             M   0 169
```

Já no conjunto de teste, temos apenas 3 de 113 (3%) tumores classificados incorretamente.

```
diagnosis_pred <- predict(treebag_fit, breast_test)
xtabs(~ diagnosis_pred + diagnosis, data = breast_test)
```

```
              diagnosis
diagnosis_pred  B  M
             B 69  1
             M  2 41
```

```
diagnosis_results <-
  tibble(prob_M = predict(treebag_fit, breast_test, type = "prob")$M,
         pred = predict(treebag_fit, breast_test, type = "raw"),
         actual = breast_test$diagnosis)
diagnosis_results
```

```
# A tibble: 113 x 3
```

```
   prob_M pred  actual
    <dbl> <fct> <chr>
 1  0.9   M     M
 2  0.933 M     M
 3  1     M     M
 4  1     M     M
 5  1     M     M
 6  1     M     M
 7  0.933 M     M
 8  0.867 M     M
 9  1     M     M
10  0     B     B
# i 103 more rows
```

Referências

- Breiman L., Friedman J. H., Olshen R. A., and Stone, C. J. (1984). **Classification and Regression Trees**. Wadsworth, Belmont.
- Breiman L. (1996). **Bagging Predictors**. Machine Learning, Boston.
- Izenman, A. J. (2008). **Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning**. Springer, New York.
- Kuhn, M. (2008). **Building Predictive Models in R Using the caret Package**. Journal of Statistical Software.
- Awan, A. A., DataCamp (2023). **What is Bagging in Machine Learning? A Guide With Examples**.